RL Chapter 4 - Dynamic Programming

Finn Ye

Feb 2025

Setup

The environment is a finite MDP with finite state S, action A, reward R, and dynamics given by probabilities $p(s', r|s, a)$.

Policy Evaluation

Given π, we can use the Bellman equation to iterate

$$v_{k+1} = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \delta v_k(s')]$$

As $k \to \infty$, $v_k \to v_\pi$.

Pseudo Code for Iterative Policy Evaluation

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(\textit{terminal})$ to 0
Loop:
$\quad \Delta \leftarrow 0$
$\quad$ **Loop for each** $s \in \mathcal{S}$:
$\quad\quad v \leftarrow V(s)$
$\quad\quad V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \delta V(s')\right]$
$\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Policy Improvement

The *policy improvement theorem* says for π, π' such that for all $s \in S$,

$$q_\pi(s, \pi'(s)) \geq v_\pi(s) \tag{1}$$

Then

$$v_{\pi'}(s) \geq v_\pi(s) \tag{2}$$

If (1) holds with strict inequality at any state, then (2) also holds strict inequality at that state.
In other worlds, if deviating to another policy only for one state is better, we have found a better policy.
Following this strategy, we can seek for improvement at every state and obtain a **greedy** policy π' that maximizes payoffs in short team.

$$\pi'(s) = \text{argmax}_a \sum_{s',r} p(s', r|s, a)[r + \delta v_\pi(s)]$$

Pseudo Code for Policy Iteration

1. **Initialization**
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$;
   $V(terminal) \doteq 0$
2. **Policy Evaluation**
   Loop:
   - $\Delta \leftarrow 0$
   - **Loop for each** $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))[r + \delta V(s')]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$
3. **Policy Improvement**
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$,
     $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \delta V(s')]$
   - **If** *old-action* $\neq \pi(s)$, **then** *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, **then** stop and return $V \approx v_*$ and $\pi \approx \pi_*$;
   else go to 2.

Value Iteration

- Policy Iteration requires sweeping through all states, which might take some time.
- Value Iteration only update every state once.

$$v_{k+1} = \max_a \sum_{s',r} p(s', r|s, a)[r + \delta v_k(s')]$$

Pseudo Code for Value Iteration

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:

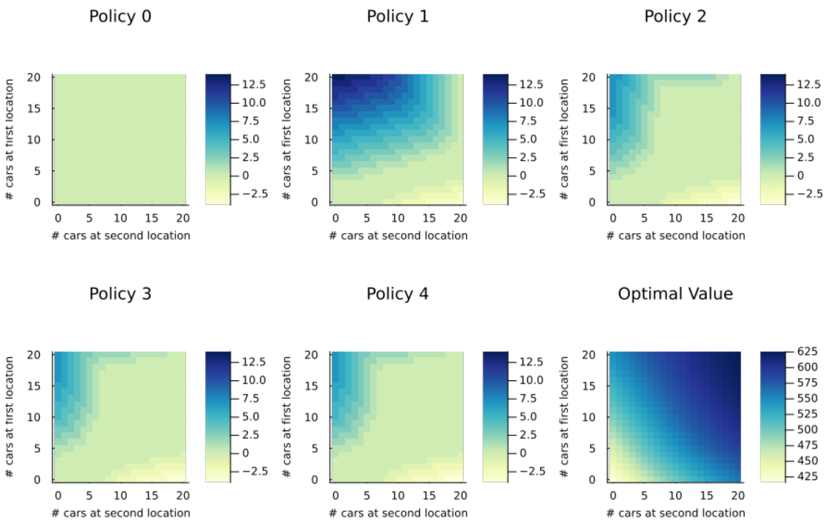
- $\Delta \leftarrow 0$
- **Loop for each** $s \in \mathcal{S}$:
  - $v \leftarrow V(s)$
  - $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a) [r + \delta V(s')]$
  - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

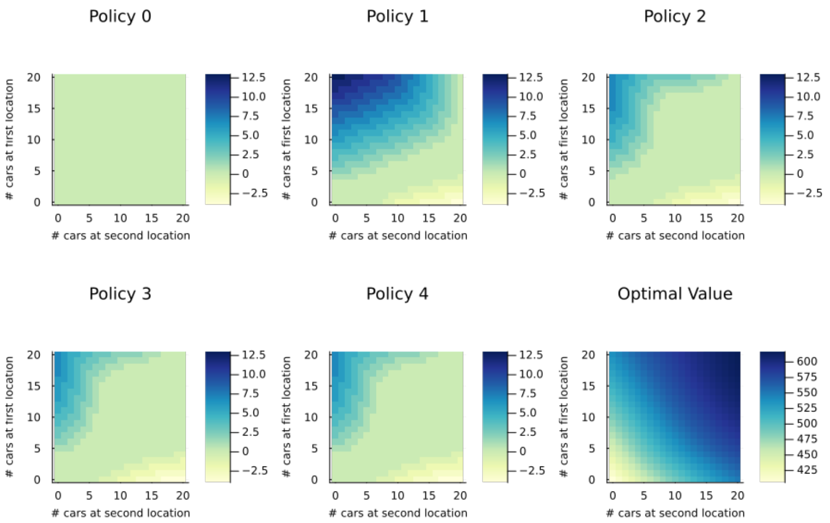
Output a deterministic policy, $\pi \approx \pi_*$, such that

$$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \delta V(s')\right]$$

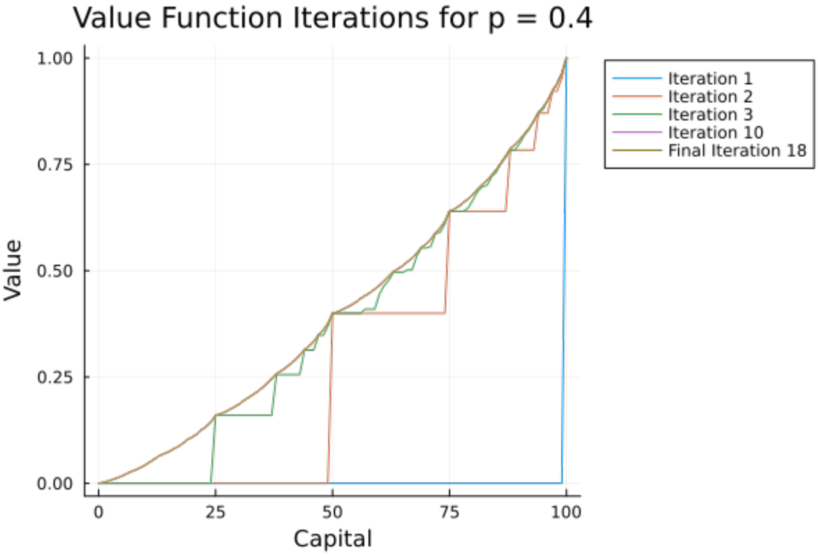
Car Rentals (Deterministic)



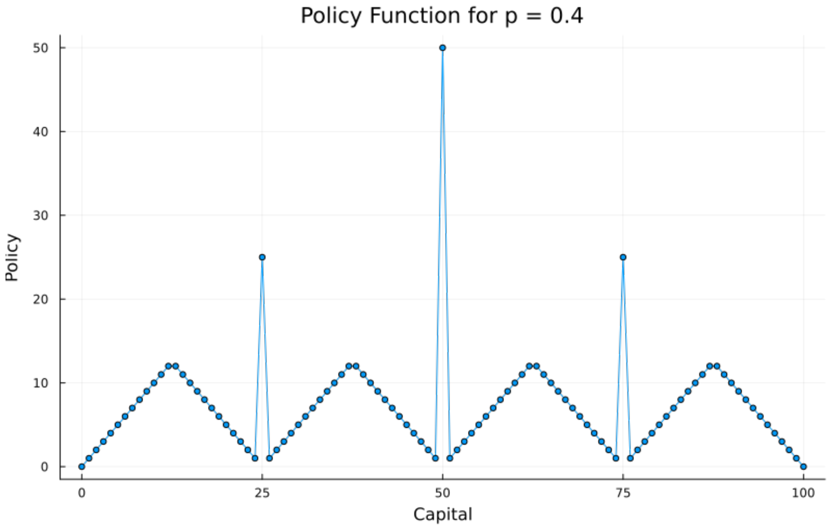
Car Rentals (Stochastic)



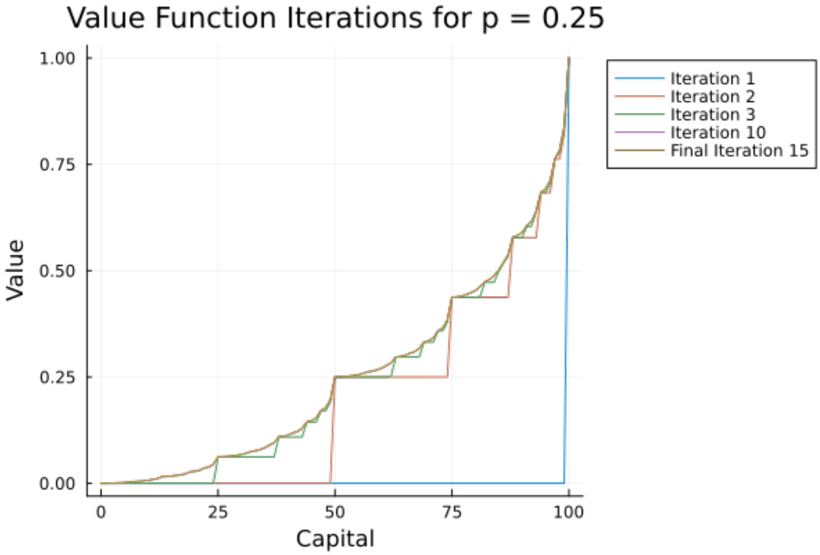
Gambler's Problem Value ($p = 0.4$)



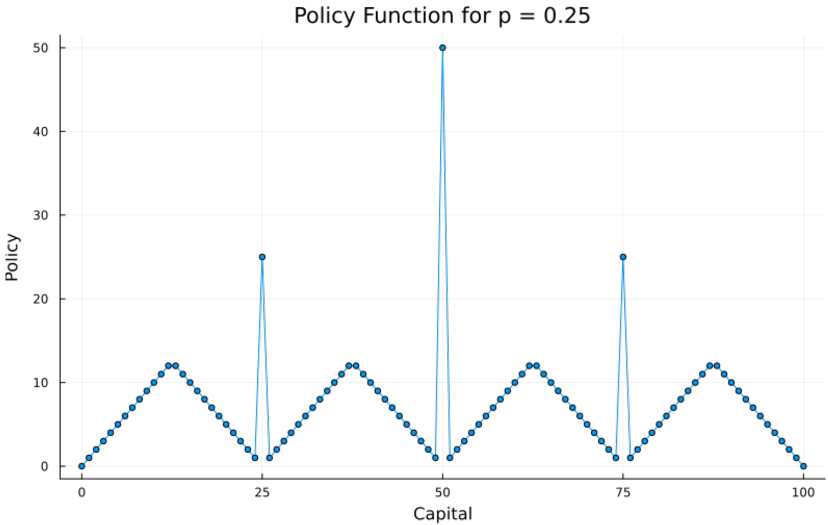
Gambler's Problem Policy ($p = 0.4$)



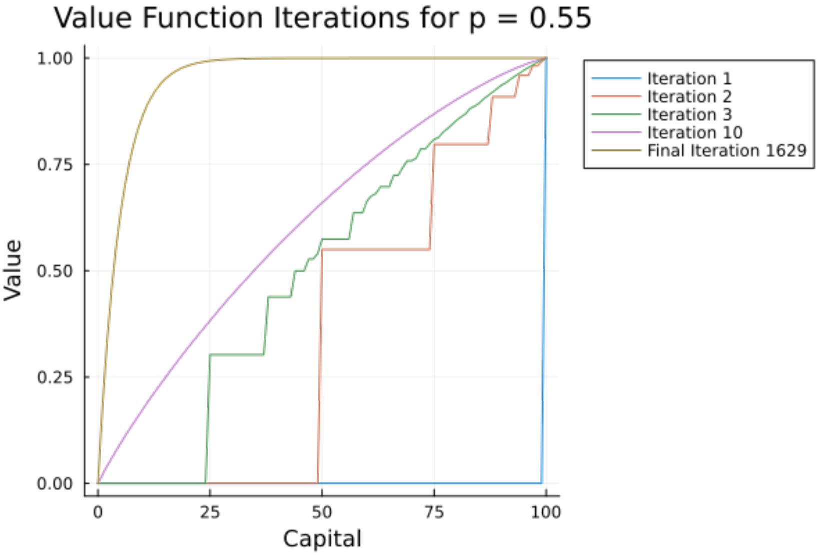
Gambler's Problem Value ($p = 0.25$)



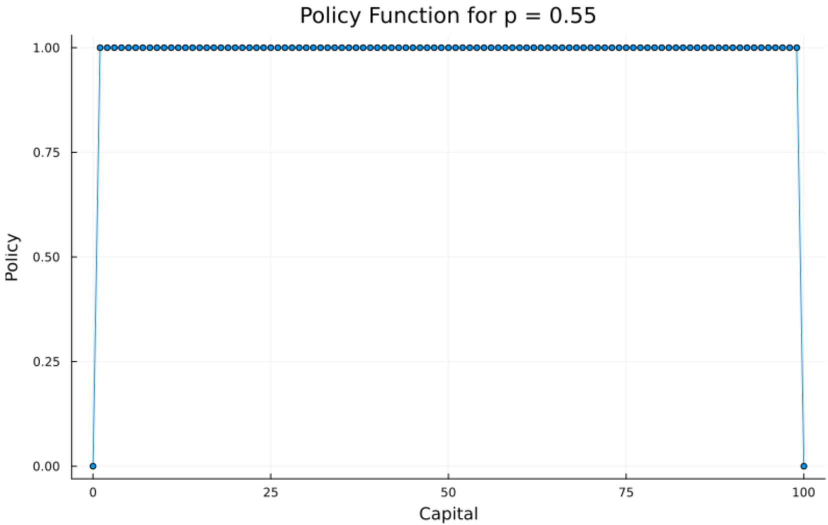
Gambler's Problem Policy ($p = 0.25$)



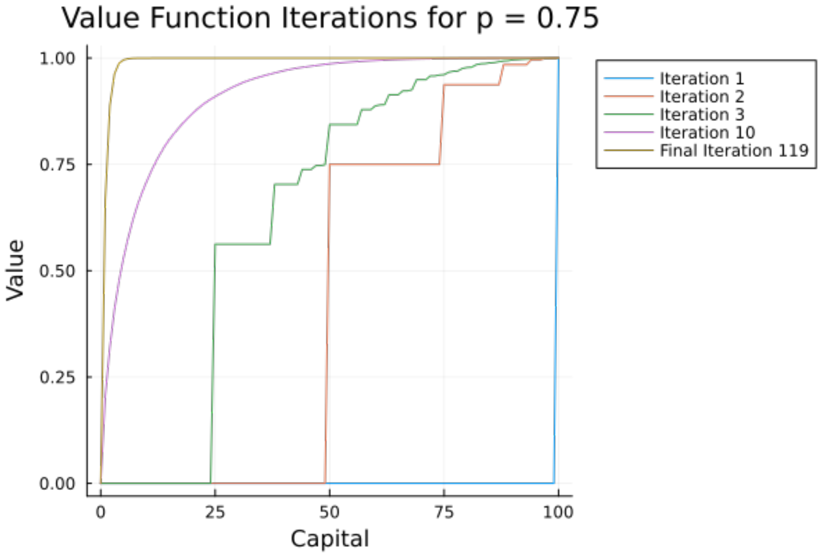
Gambler's Problem Value ($p = 0.55$)



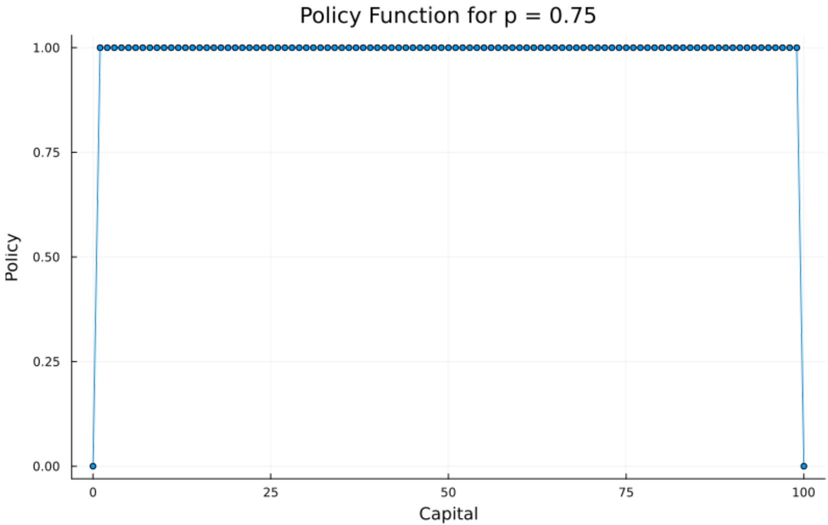
Gambler's Problem Policy ($p = 0.55$)



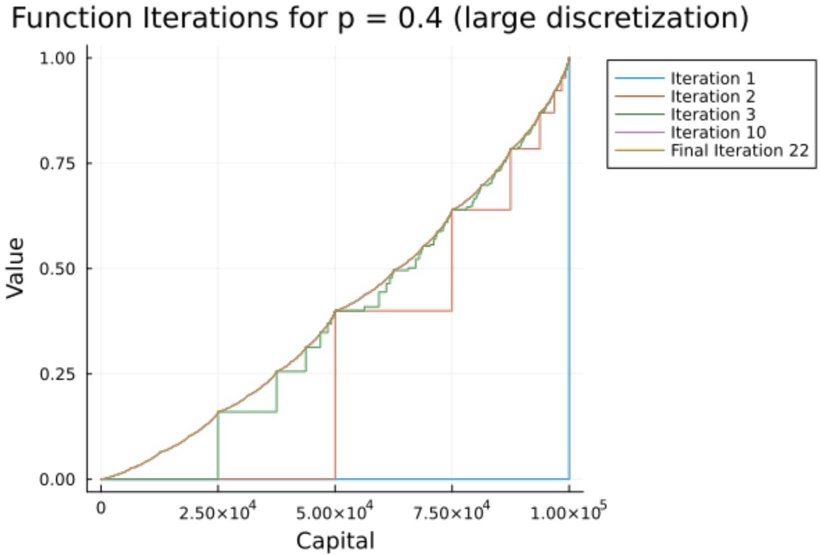
Gambler's Problem Value ($p = 0.75$)



Gambler's Problem Policy ($p = 0.75$)



Large Gambler's Problem Value ($p = 0.4$)



Large Gambler's Problem Policy ($p = 0.4$)